

# Prática da Linguagem Java

Luiz Henrique de Campos Merschmann  
Departamento de Computação Aplicada  
Universidade Federal de Lavras

luiz.hcm@ufla.br

# Na Aula de Hoje



# A Linguagem Java

## Tipos de Dados Primitivos

- ▶ *short, int, float, double, long, char, boolean e byte.*

## Declaração de Variáveis

**tipoDaVariável** nomeDaVariável;

Exemplos:

- ▶ **int** tamanho;
- ▶ **double** temperatura;
- ▶ **char** genero;

# Declaração, Atribuição e Impressão de Variáveis

```
//Declarando a variável tamanho
```

```
int tamanho;
```

```
//Atribuindo o valor 10 à variável tamanho
```

```
tamanho = 10;
```

```
//Imprimindo o conteúdo da variável tamanho
```

```
System.out.println(tamanho);
```

```
//Concatenando a saída com o operador +
```

```
System.out.println("Tamanho = " + tamanho);
```

# Observações sobre Atribuições

- ▶ Incompatibilidade na atribuição.

Exemplo:

```
double x = 5.23
```

```
int d = x;
```

Isso **não compila** e será acusado o seguinte **erro**:

*incompatible types: possible lossy conversion from double to int.*

- ▶ Situação permitida:

```
int d = 7;
```

```
double x = d;
```

Isso **compila** sem problema, uma vez que um *double* pode armazenar um número com ponto flutuante ou inteiro.

# Observações sobre Atribuições

- ▶ O código a seguir compila corretamente?

```
float a = 3.14;
```

Não. Por que?

**Resp.:** Todos os literais (constantes inseridas explicitamente no código) com ponto decimal são considerados do tipo *double* pelo Java. Como *float* não pode receber um *double* sem perda de informação, esse código não compila corretamente.

- ▶ Como fazer a atribuição acima funcionar?

```
float a = 3.14f;
```

A letra *f* indica que aquele literal deve ser tratado como um *float*.

# Observações sobre Atribuições

## *Casting*

- ▶ Em algumas situações precisamos que um valor em ponto flutuante seja **truncado** e armazenado em uma variável inteira. Para não haver erro de compilação, é preciso ordenar que o valor em ponto flutuante seja moldado (casted) como um número inteiro.

Exemplo:

```
double x = 1.23;
```

```
int i = (int) x;
```

# Controle de Fluxo

## If

```
if (condicaoBooleana) {  
    código;  
}
```

onde *condicaoBooleana* é qualquer expressão que retorne *verdadeiro* ou *falso*.

## Exemplo:

```
if (tamanho < 1.20) {  
    System.out.println("Não pode passar");  
}
```



# Controle de Fluxo

## If \ Else

```
if (condicaoBooleana) {  
    código;  
} else {  
    outro código;  
}
```

onde *condicaoBooleana* é qualquer expressão que retorne *verdadeiro* ou *falso*.

### Exemplo:

```
if (tamanho < 1.20) {  
    System.out.println("Não pode passar");  
} else {  
    System.out.println("Pode passar");  
}
```

# Controle de Fluxo

## If \ Else If \ Else

```
if (condicaoBooleana1) {  
    código1;  
} else if (condicaoBooleana2) {  
    código2;  
} else {  
    código3;  
}
```

### Exemplo:

```
if (nota >= 60) {  
    System.out.println("Aprovado");  
} else if (nota >= 50) {  
    System.out.println("Reprovado – direito a turma Z");  
} else {  
    System.out.println("Reprovado – sem direito a turma Z");  
}
```

# Controle de Fluxo

## While

```
while (condicaoBooleana) {  
    código;  
}
```

onde *condicaoBooleana* é qualquer expressão que retorne *verdadeiro* ou *falso*.

### Exemplo:

```
i = 50;  
while (i < 100) {  
    System.out.println(i);  
    i+=1;  
}
```

# Controle de Fluxo

## For

```
for (inicialização; condição; incremento) {  
    código; // executa enquanto a condição for verdadeira.  
}
```

### Exemplo:

```
for (int i = 50; i < 100; i++) {  
    System.out.println(i);  
}
```

# Controle de Fluxo

## For - Controlando o *loop*

- ▶ Mesmo tendo a condição booleana em nossos laços (for), em algumas situações podemos desejar parar o *loop* antes da condição booleana se tornar *falsa*.
- ▶ Para isso utilizaremos o comando *break*.

### Exemplo:

```
for (int i = x; i < y; i++) {  
    if (i % 19 == 0) {  
        System.out.println("Achei!");  
        break;  
    }  
}
```

O que esse código faz?

Encontra o primeiro número divisível por 19 entre x e y.

# Controle de Fluxo

## Do \ While

```
do {  
    código; // executa enquanto a condicao for verdadeira;  
} while(condicao);
```

- Observe que agora o teste da condição de parada é feito no fim. Desse modo, esse laço é executado pelo menos uma vez.

### Exemplo:

```
char opcao = lerOpcao();  
do {  
    código; // faça algo com base na opção escolhida  
    opcao = lerOpcao();  
} while (opcao != 'S');
```

# Controle de Fluxo

## Switch

```
switch (expressao) {  
    case a:  
        código do case a;  
    case b:  
        código do case b;  
    case c:  
        código do case c;  
    default:  
        código do default;  
}
```

onde:

- ▶ *expressao* deve ser do tipo *byte*, *short*, *int*, *char* ou *String*.
- ▶ cada **case** deve conter uma expressão integral constante, ou seja, um valor constante do tipo *byte*, *short*, *int*, *char* ou *String* (a expressão também pode ser uma variável constante – *final*).

# Controle de Fluxo

## Switch

### Exemplo:

```
int x;  
:  
switch (x/10) {  
    case 6:  
        código do case 6;  
    case 7:  
        código do case 7;  
    case 8:  
        código do case 8;  
    default:  
        código do default;  
}
```



# Arrays

- ▶ Objetos *array* são estruturas de dados consistindo em itens de dados do mesmo tipo.
- ▶ *Arrays* são úteis quando se deseja processar grupos de valores relacionados.  
Exemplo: conjunto de notas de alunos.

The diagram illustrates an array structure. On the left, the text 'Nome do array (c)' has an arrow pointing to the first row of the table. Below it, the text 'Índice (ou subscripto) do elemento no array c' has an arrow pointing to the index '11' in the last row of the table.

c[ 0 ]	-45
c[ 1 ]	6
c[ 2 ]	0
c[ 3 ]	72
c[ 4 ]	1543
c[ 5 ]	-89
c[ 6 ]	0
c[ 7 ]	62
c[ 8 ]	-3
c[ 9 ]	1
c[ 10 ]	6453
c[ 11 ]	78

# Declaração, Criação e Inicialização de *Arrays*

- ▶ Para criar um *array*, especifique o tipo dos elementos do *array* e o número de elementos<sup>1</sup>.

## **Exemplo:**

```
int[ ] c; //declara a variável de array
```

```
c = new int[10]; //cria o array
```

OU

```
int[ ] c = new int[10];
```

Criando vários arrays em uma única declaração:

```
String[ ] a = new String[10], b = new String[50];
```

- ▶ Inicialização de um *array*

## **Exemplo:**

```
int[ ] c = {32, 45, 21, 6, 85, 4};
```

---

<sup>1</sup>O tamanho dos *arrays* permanece fixo depois que eles são criados.

# Exemplo que Utiliza *Array*

```
public class ManipularArray{
    public static void main(String[] args){
        int[] v = {23,34,45,56};

        //Título das colunas
        System.out.printf("%s%8s%n", "Índice", "Valor");

        //Imprimindo o valor de cada elemento do array
        for(int cont = 0; cont < v.length; cont++){
            System.out.printf("%5d%8d%n", cont, v[cont]);
        }
    }
}
```

Saída:

Índice	Valor
0	23
1	34
2	45
3	56

# Perguntas?

